



Vendedor Puerta-en-puerta

Laboratorio Semana 4

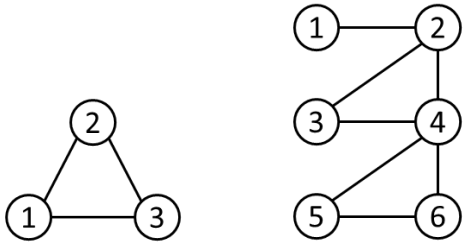
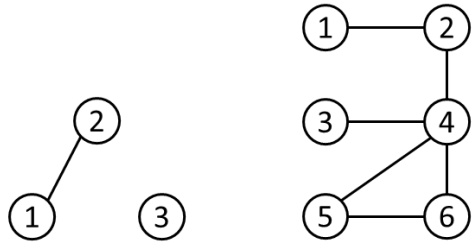
1 Introducción

Debido a los altos costos de la gasolina que importan de Venezuela, las compañías de vendedores puerta-en-puerta Amway, Avon, y Herbalife han decidido organizarse. Para cada vendedor existe un grafo no-dirigido donde los nodos representan las puertas de las compradoras, y las aristas representan si existe una calle transitable a pie entre ellas. Usted ha sido contratado para desarrollar un programa que indique, al dársele uno de estos grafos, si es posible para ese vendedor visitar a todas sus compradoras a pie de forma eficiente (sin tener que pasar dos veces frente a la misma casa) o si necesita usar un automóvil.

El dueño de Amway leyó, en una revista de la Academia Nacional de Ciencias, que el mejor camino para hacer tales recorridos se denomina Camino Hamiltoniano y desea que el camino encontrado sea de este tipo. Si no, debe concluirse que ese vendedor no puede visitar a sus compradoras a pie. En aras de plegarse a la nueva tendencia de Inteligencia Explicable, el programa debe mostrar cómo llegó a esa conclusión.

2 Descripción de la Actividad

Realice un programa que indique si existe un Camino Hamiltoniano para un grafo no-dirigido dado. Un Camino Hamiltoniano es un camino elemental (es decir, un camino donde cada vértice aparece una sola vez) que pasa por todos los nodos del grafo 1 vez. Puede encontrar más información en la sección 17.3 de libro [Algorithms in Java](#) de Robert Sedgewick.

| | |
|---|--|
|  |  |
| Ejemplos de grafos donde sí existe un Camino Hamiltoniano | Ejemplos de grafos donde no existe un Camino Hamiltoniano |

Si existe un camino, debe indicar

- cuál es ese camino
- cuántos vértices contiene
- qué recorrido siguió para hallarlo

Si no existe el camino, igualmente debe mostrar el recorrido seguido para llegar a esta conclusión.

El programa debe poder recorrer el grafo tanto usando *Breadth-First Search* como usando *Depth-First Search*. Debe retornar el primer Camino Hamiltoniano que encuentre.

2.1 Sugerencias

Para hacer la visualización más cómoda, modifique la clase GraphPath.java para agregar una variable `depth` que se incremente al entrar en la recursión y se decremente al salir de la recursión, y luego añadir al inicio del método recursivo código necesario para imprimir tantos espacios como lo indique la variable `depth` seguido por la información necesaria.

Pruebe un camino que comienza desde cada uno de los vértices. El vértice de comienzo para realizar el recorrido (BFS/DFS) puede ser escogido absolutamente de cualquier manera, siempre y cuando no se elija el mismo vértice de comienzo dos veces.

3 Entrada de Datos

El programa se debe poder ejecutar desde la consola con el siguiente comando:

```
java Hamilton <archivo> <BFS|DFS>
```

donde `<archivo>` es la ruta del archivo de grafo que se desea traducir, y `<BFS|DFS>` son las siglas “BFS” ó “DFS” indicando el algoritmo con el cual se debe recorrer el grafo.

3.1 Formato de archivo

Los grafos se componen de una lista de aristas en el formato

```
n
m
ni1 nf1
ni2 nf2
:
nim nfm
```

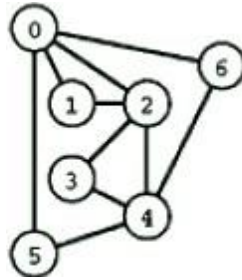
donde

- n es el número de nodos
- m es el número de aristas
- ni_i es el índice del vértice ($0 \leq ni_i < n$) al inicio de la i -ésima arista
- nf_i es el índice del vértice ($0 \leq nf_i < n$) al final de la i -ésima arista

Si una arista aparece dos veces (por ejemplo, si en el archivo aparece la arista 2-3 y la arista 3-2 como aristas distintas) debe lanzarse una excepción

4 Salida de ejemplo

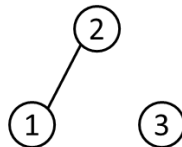
Suponga el siguiente grafo de entrada:



```
> java Hamilton ejemploConCamino.txt DFS
Recorrido desde 6:
  6-0
    0-1
      1-2
        2-0 ya visitado
        2-3
          3-4
            4-2 ya visitado
            4-5
Camino encontrado: 6-0-1-2-3-4-5
El camino tiene 7 vértices
```

Observe que se ha elegido el nodo 6 arbitrariamente como punto de inicio y, casualmente, produjo un recorrido válido.

Suponga, en cambio, el siguiente grafo de entrada:



```
> java Hamilton ejemploSinCamino.txt BFS
Recorrido desde 1:
  1-2
Recorrido desde 2:
  2-1
Recorrido desde 3:
No existe camino
```

Observe que se probó iniciar el recorrido desde todos los nodos.

5 Entrega

Debe entregar, a través del Moodle antes del jueves, 6 de febrero a la 1:00 pm., como mínimo

- Una clase `Grafo`
- Una clase `DFS` que contenga el código para recorrer el grafo y almacene el camino encontrado
- Una clase `BFS` que contenga el código para recorrer el grafo y almacene el camino encontrado
- Una clase `Hamilton` que contenga el `main`

6 Evaluación

De ser escogido para evaluación, se asignarán:

- 5 puntos por código
 - 2 puntos por implementar correctamente BFS
 - 2 puntos por implementar correctamente DFS
 - 1 punto por la carga del archivo
- 5 puntos por ejecución
 - 2 puntos por identificar correctamente cuando sí existe un Camino Hamiltoniano
 - 1 punto con DFS
 - 1 punto con BFS
 - 2 puntos por identificar correctamente cuando no existe un Camino Hamiltoniano
 - 1 punto con DFS
 - 1 punto con BFS
 - 1 punto por correctamente dar error al encontrar una arista repetida

Se asignará un punto adicional si el programa es capaz de detectar Ciclos Hamiltonianos. Debe agregar un tercer parámetro a la llamada desde consola que indique si se deben buscar ciclos. Si el tercer parámetro es omitido, debe suponerse que se desean buscar Caminos Hamiltonianos.